PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H03B 29/00	A2	(11) International Publication Number: WO 98/48508
		(43) International Publication Date: 29 October 1998 (29.10.98)

(21) International Application Number: PCT/US98/07920

(22) International Filing Date: 17 April 1998 (17.04.98)

(30) Priority Data:
60/044,673 18 April 1997 (18.04.97) US

(71) Applicant: UNIVERSITY OF UTAH RESEARCH FOUNDATION [US/US]; Suite #170, 421 Wakara Way, Salt Lake City, UT 84108 (US).

(72) Inventor: DOUGLAS, Scott, C.; 130 South 1300 East #706, Salt Lake City, UT 84102 (US).

(74) Agents: O'BRYANT, David, O. et al.; Thorpe, North & Western, LLP, P.O. Box 1219, Sandy, UT 84091-1219 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: METHOD AND APPARATUS FOR MULTICHANNEL ACTIVE NOISE AND VIBRATION CONTROL

(57) Abstract

A method and apparatus for implementing a fast, exact implementation of a filtered–X LMS adaptive filter for which the system's complexity scales according to the number of filter coefficients within the system. A mathematically equivalent controller system intentionally introduces a delay in a feedback signal to thereby reduce the number of calculations necessary to calculate coefficients. The system then utilizes delay compensation to overcome the effects of the delay, resulting in a system which generates the same results as a more costly and computationally more intensive controller system.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

METHOD AND APPARATUS FOR MULTICHANNEL ACTIVE NOISE AND VIBRATION CONTROL

BACKGROUND

The present invention was at least partially funded by DARPA Grant No. DAAH04-96-1-0085

1. Field of the Invention

The present invention relates to a method and apparatus for improved active noise and vibration control. More specifically, intentional delay is introduced into a feedback loop to thereby reduce complexity of the equations, and then the invention compensates for the intentionally induced delay to thereby realize the benefits of the reduced complexity of the system.

2. The State Of The Art

In active noise and vibration control, undesired sound in an acoustic region is attenuated by superimposing an equal-but-opposite acoustical field in the region. The "anti-noise" is created by measuring in real time the source of the unwanted noise with input sensors, processing this information digitally to produce output signals, and sending these signals to the desired quiet region using actuators. Additional error sensors are placed in the quieted regions to provide feedback for a control system to adjust its characteristics.

Active noise and vibration control (ANVC) systems are known in the prior art. In particular, the filtered-X LMS (least mean square) algorithm has shown promise, but has been difficult and often costly to

implement because it is too computationally complex. Nevertheless, the algorithm has several advantages. First, it is well-suited to both broadband and narrowband control tasks, with a structure than can be adjusted according to the problem at hand. Second, it is easily described and understood, especially given the vast background literature on adaptive filters upon which the algorithm is based. Third, its structure and operation are ideally suited to the architecture of standard digital signal processing (DSP) chips, due to the algorithm's extensive use of the multiply/accumulate (MAC) operation. Fourth, the algorithm behaves robustly in the presence of physical modeling errors and numerical precision effects caused by finite-precision calculations. Fifth, the algorithm is relatively simple to set up and tune in a real-world environment.

Briefly, it is mentioned that the filtered-X algorithm is a modification of the basic feedforward LMS algorithm. The modification is required when dynamics exist between the adaptive filter output and the error sensor signals. The modification to the update equation includes an estimate of the plant transfer function.

Despite its advantages, the filtered-X LMS algorithm suffers from at least one drawback that makes it difficult to implement when a multichannel controller is desired: the complexity of the coefficient updates for the finite-impulse-response (FIR) filters within the controller in this situation is often much greater than the complexity of the input-output calculations. It is not unusual for the coefficient updates to require more than ten times the number of MACs needed to compute the outputs of the controller for fixed coefficient values, and the

situation worsens as the number of error sensors is increased. For this reason, recent efforts have focused on ways to reduce the complexity of the filtered-X LMS algorithm in a multi-channel context.

Some of the suggested changes to reduce complexity include: (1) block processing of the coefficient updates using fast convolution techniques, (2) partial updating of the controller coefficients, and (3) filtered-error methods. While useful, these methods often reduce the overall convergence performance of the controller, either because they introduce additional delays into the coefficient update loop or because they throw away useful information about the state of the control system. Such a performance loss may not be tolerable in some applications.

In addition to these computational difficulties, the multichannel filtered-X LMS algorithm also suffers from excessive data storage requirements. This algorithm employs filtered input signal values that are created by filtering every input signal by every output-actuator-to-error-sensor channel of the acoustic plant. The number of these terms can be an order-of-magnitude greater than the number of controller coefficients and input signal values used in the input-output calculations. As typical DSP chips have limited on-chip memory, system designers may be forced to use costly off-chip memory within their controller architectures that can further slow the operation of the system due to limits in input/output data throughput. While some of the aforementioned techniques for complexity reduction also have reduced memory requirements, the performance of the overall system is effectively limited by these methods.

It would be an advantage over the state of the art to provide a fast, exact implementation of a multichannel system. It would also be an improvement to reduce delays of secondary paths within the coefficient updates. It would be especially advantageous to be able to generate the same output signals using mathematically equivalent but less complex equations which would accordingly require less hardware to implement. The method should preserve the characteristic robust and accurate behavior of the filtered-X LMS algorithm, especially for controllers having a large number of channels

OBJECTS AND SUMMARY OF THE INVENTION

It is an object of the present invention to provide an improved method and apparatus for active noise and vibration control for multichannel systems.

It is another object to provide a method and apparatus for ANVC which intentionally introduces a delay within a set of operations to reduce system complexity.

It is another object to provide a method and apparatus for ANVC which utilizes delay compensation to implement a system which is mathematically equivalent to but much less complex than existing ANVC systems.

It is another object to provide an ANVC system that intentionally delays a feedback signal within the system, and then compensates for the delay as seen at an output end of the control system.

It is another object to provide a ANVC system in which the delay compensation technique can correct for delays caused by the propagation of acoustic energy through an output-actuator-to-error-sensor portion of the

system to thereby eliminate a systematic performance loss that was introduced when the filtered-X LMS control algorithm was heuristically derived from the LMS algorithm.

In the original implementation of the filtered-X LMS algorithm, the sensitivity of the controller's performance to the feedback error signals is propagated back to the controller states through a brute-force calculation step. The present invention accumulate the sensitivities of the error signals to the state of the controller in such a way as to avoid this costly recombination step with its order-of-magnitude increase in the number of instructions per sample time.

The present invention is realized in a method and apparatus for implementing a fast, exact implementation of a filtered-X LMS adaptive filter for which the system's complexity scales according to the number of filter coefficients within the system. A mathematically equivalent controller system intentionally introduces a delay in a feedback signal to thereby reduce the number of calculations necessary to calculate coefficients. The system then utilizes delay compensation to overcome the effects of the delay, resulting in an system which generates the same results as a more costly and computationally more intensive controller system.

In a first aspect of the invention, calculations associated with the feedback portion of a signal recombination step which were originally computationally-costly are eliminated. In the present invention, the sensitivities of the error signals to the state of the controller are accumulated in such as way as to avoid this costly recombination step through a separation of

the computations.

In another aspect, the delay compensation technique in a single channel system is useful where the output actuators are located close to the error signals and far away from the input sensors.

In another aspect of the invention, the separability of the computations makes the new algorithm beneficial in all multichannel controller systems except those with only a coupled of controller channels.

These and other objects, features, advantages and alternative aspects of the present invention will become apparent to those skilled in the art from a consideration of the following detailed description taken in combination with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a prior art single-channel filtered-X LMS adaptive controller.

Figure 2 is a block diagram of a preferred embodiment which is made in accordance with the principles of the present invention, and shows a single-channel LMS adaptive controller employing delay compensation.

Figure 3A is a simulated performance of the present invention on air compressor noise when there is an unattenuated noise power.

Figure 3B is a simulated performance of the present invention on air compressor noise when there is residual noise power for the adjoint LMS/CPFE (corrected phase filtered error), fast filtered-X LMS, and fast LMS algorithms.

Figure 3C is a simulated performance of the present

invention on air compressor noise when there is residual noise power for the fast filtered-X LMS and fast LMS algorithms with stabilization.

DETAILED DESCRIPTION OF THE INVENTION

Reference will now be made to the drawings in which the various elements of the present invention will be given numerical designations and in which the invention will be discussed so as to enable one skilled in the art to make and use the invention. It is to be understood that the following description is only exemplary of the principles of the present invention, and should not be viewed as narrowing the claims which follow.

The present invention is a fast, exact implementation of an adaptive filter for which the system's complexity scales according to the number of filter coefficients within the system. Furthermore, the present invention extends computationally-efficient methods for effectively removing the delays of the secondary paths within the coefficient updates. The present invention accomplishes these tasks using algorithms which are mathematically equivalent to the original filtered-X LMS algorithm.

To distinguish the present invention from the prior art filtered-X LMS algorithm, it is necessary to examine the equations of the prior method to determine how the method has changed. To simplify the explanation, the single-channel filtered-X LMS adaptive feedforward controller is presented. The extension to the multi-channel controller will then be explained in the context of the single-controller.

Figure 1 shows a block diagram of a controller

system implementing a single-channel filtered-X LMS algorithm. A sensor is placed near a noise source to collectb samples of the input signal $x(n)$ for processing by the system. This system computes an actuator output signal $y(n)$ using a time-varying FIR filter of the form:

Equation (1): $$y(n) = \sum_{l=0}^{L-1} w_l(n)x(n-l),$$

where $w_l(n), 0 \le l \le L-1$ are the controller coefficients and L is the controller filter length. The acoustic output signal produced by the controller combines with the noise as it propagates to the quiet region, where an error sensor collects the combined signal. This error is modeled as

Equation (2): $$\epsilon(n) = d(n) + \sum_{m=-\infty}^{\infty} h_m y(n-m),$$

where $d(n)$ is the unattenuated noise signal and $h_m, -\infty < m < \infty$ is the plant impulse response. Note that equation (2) is never computed because $\epsilon(n)$ is a measurement of a physical quantity.

Referring to equation (1), this is the input-output relationship for the controller. The controller computes y(n) to create anti-noise or anti-vibration. The calculation represented by equation (1) must be performed at each sample time. In the present invention, this equation is effectively not reduced, but is the same as the prior art algorithm.

Referring to equation (2), this equation describes a physical process which occurs. Consequently, equation (2) mathematically defines an accurate model. Therefore, this is something which is not computed, it is just

measured. The quantity is measured by the error microphone or error sensors. Therefore, the result of equation (2) is the input from the error sensors.

The filtered-X LMS coefficient updates are given by Equation (3):

$$w_l(n+1) = w_l(n) - \mu(n)\epsilon(n)f(n-l),$$

where the filtered input sequence $f(n)$ is computed as

Equation (4):

$$f(n) = \sum_{m=1}^{M} h_m x(n-m),$$

and M is the FIR filter length of an appropriate estimate of the plant impulse response. In practice, the values of h_m used in equation (4) are estimates of the actual h_m in equation (2) and are usually obtained in a separate estimation procedure that is performed prior to the application of control.

Referring to equation (3), this equation will be different from the prior art filtered-X LMS algorithm. It describes how the quantity w is being changed by way of coefficients (parameters, numerical values stored in memory). It is necessary to store l parameters, where l represents filter length. These values for w are calculated for each sample time. Where the quantities in equation (3) are fixed numbers or measured values, the quantity being calculated is f(n-1).

Looking at equation (4), this equation is very similar to the input-output equation of the controller introduced in equation (1). The parameter h_m must be performed at each sample time.

A study of equations (1), (3) and (4) shows that the filtered-X LMS algorithm requires $2L + M + 1$ multiply/accumulate (MAC) operations and

$2L + \max\{L, M+1\} + 1$ memory locations to store the necessary $w_l(n), h_m, x(n-1)$, and $f(n-1)$ for the algorithm at each step. For typical choices of the controller and plant filter lengths, the complexity and memory requirements of this algorithm are reasonable. As will be shown, however, such is not the case for the natural extension of this algorithm to the multichannel control task.

The new algorithm is essentially a change in the way that equations (3) and (4) are calculated. Most importantly, the present invention does not calculate equation (4) at all. Consequently, there are realized substantial savings in memory space because of the elimination of the calculations required to obtain the result of equation (4).

The present invention essentially changes the way that the result of equation (3) is updated. This in turn affects the way in which equation (1) is calculated.

The present invention will now be shown in this embodiment as a modified implementation of the single channel filtered-X LMS algorithm. This method combines the adjoint LMS/corrected phase filtered error (CPFE) algorithm with a method for delay compensation used in fast projection adaptive filters. To derive the implementation, the coefficient updates of the original algorithm are in the form of

Equation (5):

$$w_l(n+1) = w_l(n) - \mu(n)\epsilon(n)\sum_{m=1}^{M} h_m x(n-l-m).$$

Define $\epsilon_m(n)$ for $1 \le m \le M$ as

Equation (6):

$$\varepsilon_m(n) = h_m \mu(n)\epsilon(n).$$

Then, equation (5) becomes

Equation (7): $$w_l(n+1) = w_l(n) - \sum_{m=1}^{M} \varepsilon_m(n)x(n-l-m).$$

The relation in equation (7) for M successive time steps is represented as

Equation (8): $$w_l(n+1) = w_l(n-M+1) - \sum_{p=0}^{M-1}\sum_{m=1}^{M} \varepsilon_m(n-p)x(n-l-m-p).$$

The summation on the right-hand-side of equation (8) is expanded in a particularly useful way as Equation (9):

$$\begin{aligned} w_l(n+1) = \hat{w}_l(n) \\ - \varepsilon_1(n)x(n-l-1) &- \varepsilon_2(n)x(n-l-2) \cdots - \varepsilon_M(n)x(n-l-M) \\ &- \varepsilon_1(n-1)x(n-l-2) \cdots - \varepsilon_{M-1}(n-1)x(n-l-M) \\ &\ddots \qquad\qquad \vdots \\ &\qquad - \varepsilon_1(n-M+1)x(n-l-M) \end{aligned}$$

where the lth auxiliary coefficient $w_l(n)$ is defined as

Equation (10): $$\hat{w}_l(n) = w_l(n-M+1) - \sum_{p=1}^{M-1}\sum_{m=p+1}^{M} \varepsilon_m(n-M+m-p)x(n-l-M-p).$$

The expression in equation (9) indicates an important fact about the structure of the filtered-X LMS updates: the same input sample $x(n-l-m)$ is used in successive time instants to update the same coefficient $w_l(n)$. This structure is exploited to develop a set of coefficient updates that are grouped according to the individual $x(n-l-m)$ values appearing on the right-hand-side of equation (9). Such a scheme updates the lth auxiliary coefficient $\hat{w}_l(n)$ rather than the actual controller coefficient $w_l(n)$. Define $e_m(n)$ as

Equation (11): $$e_m(n) = \sum_{p=1}^{m} \varepsilon_p(n-m+p).$$

Then, it is straightforward to show that $\hat{w}_l(n)$ can be updated as

Equation (12):

$$\hat{w}_l(n+1) = \hat{w}_l(n) - e_M(n)x(n-M-l).$$

Thus, $\hat{w}_I(n+1)$ is obtained by subtracting from $\hat{w}_I(n)$ the last column of terms on the right-hand-side (RHS) of equation (9).

Equation (12) is essentially a replacement for equation (3) of the prior art. Equation (12) basically updates coefficients in a very different manner by multiplying x by a quantity e_M. The most significant difference then is that new equation (12) does not use $f(n)$ in the updating. Consequently, $f(n)$ no longer needs to be computed, thus eliminating the computation of equation (4). What does need to be computed is the new value e_M. The quantity e_M is defined in equation (11). Looking at equation (6), the result of equation (11) is defined, again, in terms of h_m. Therefore, the algorithm proceeds where the sequence of calculations performed is equation (6), then equation (11), then equation (12). In other words, equations (6), (11) and (12) replace equations (3) and (4).

A problem is that a different update quantity is the result in equation (12). Therefore, y is now calculated in a new way. The new equation for y is shown in equation (16). The first summation is similar to the original term in equation (1). However, the second summation is a new term which is defined as the delay compensation term. The terms of the second summation are defined in equations (17) and (18).

Since $e_M(n)$ is obtained by filtering $\mu(n)\epsilon(n)$ by the time-reversed plant impulse response $\{h_M, h_{M-1}, \ldots, h_1\}$, equation (12) is the single-channel version of the adjoint LMS/CPFE algorithm. What is novel is the relationship in equation (9) that provides the link

between $\hat{w}_l(n)$ and $w_l(n)$, or, equivalently, the link between the adjoint LMS/CPFE and filtered-X LMS algorithms. Equation (9) is used to compute $y(n)$ for the filtered-X LMS algorithm using $\hat{w}_l(n)$ as calculated by equation (12). To proceed, the expression for $w_l(n+1)$ is substituted in equation (7) into equation (9). Using equation (11), the equation obtained is

Equation (13): $$w_l(n) = \hat{w}_l(n) - \sum_{m=1}^{M-1} e_m(n-1)x(n-l-m-1).$$

Substituting the expression for $w_l(n)$ in equation (13) into equation (1), the equivalent expression is produced as Equation 14:

$$y(n) = \sum_{l=0}^{L-1} \hat{w}_l(n)x(n-l) - \sum_{l=0}^{L-1}\sum_{m=1}^{M-1} e_m(n-1)x(n-l-m-1)x(n-l).$$

Define the correlation term $r_m(n)$ as

Equation (15): $$r_m(n) = \sum_{l=0}^{L-1} x(n-l-m)x(n-l).$$

Then, equation (14) becomes

Equation (16): $$y(n) = \sum_{l=0}^{L-1} \hat{w}_l(n)x(n-l) - \sum_{m=1}^{M-1} e_m(n-1)r_{m+1}(n).$$

Such a calculation is of reasonable complexity because $r_m(n)$ can be recursively updated as

Equation (17): $$r_m(n) = r_m(n-1) + x(n)x(n-m) - x(n-L)x(n-L-m).$$

Moreover, $e_m(n)$ has a simple order-recursive update of the form

Equation (18): $$e_m(n) = \begin{cases} h_1\epsilon_\mu(n) & \text{if } m = 1 \\ e_{m-1}(n-1) + h_m\epsilon_\mu(n) & \text{if } 2 \le m \le M \end{cases},$$

where

Equation (19): $\epsilon_{\mu}(n) = \mu(n)\epsilon(n)$.

Collecting (12), (16), (17), (18), and (19), a set of equations is obtained that exactly computes the output signal of the filtered-X LMS adaptive controller. This algorithm requires $2L+4M-1$ MACs to implement at each iteration. Thus, this version is more computationally-complex than the original implementation of the filtered-X LMS algorithm, which only requires $2L+M+1$ MACs per iteration. In the multichannel case, however, the alternative implementation can save operations and memory storage, as is now show.

It was also recognized that equation (11) could be simplified. This new equation is shown in equations (18) and (19).

Therefore, the new algorithm is defined in terms of new equations (12), (16), (17), (18) and (19).

It is important to the present invention to realize that delay is intentionally introduced which is of surprising benefit to the invention. Of course, equation (16) thus now includes the delay compensation term. The delay introduced into the algorithm is introduced in equation (12) in the term of $e_M(n)$. This term is a filtered or delayed version (at different values of frequency or amplitude) of the original signal epsilon. However, the result of equation (12) is not what is wanted. Nevertheless, it is possible to compensate for the delay through equation (16) and still obtain the output. Therefore, the result of equation (16), y(n), is mathematically identical to the y(n) of equation (1) of the prior art. Numerically, they will of course be slightly different because of numerical inaccuracies

introduced in calculations because of the precision of the computing device. Consequently, the same result is obtained through different calculations. It is at this point that the computational efficiencies of the present invention become apparent when viewed for the multichannel scenario. The more channels which are utilized in obtaining the data, the more calculations must be accomplished. Therefore, although the efficiencies of the present invention are still applicable to the single channel scenario, they are amplified in the multichannel scenario. The multichannel complexity comparison and savings of the present invention are made explicit in table 3 of the same document.

Having described the single channel controller and the development of the algorithms thereof, the multichannel version of the filtered-X LMS algorithm in its original implementation will now be described. Then the preferred embodiment of the present invention will be presented for the multichannel scenario.

In multichannel control, I input sensors are used to collect I input signals $x^{(i)}(n)$, $1 \le i \le I$. The controller computes J output signals $y^{(j)}(n)$, $1 \le j \le J$, as

Equation (20): $$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} w_l^{(i,j)}(n) x^{(i)}(n-l),$$

where $w_l^{(i,j)}(n)$, $0 \le l \le L-1$ are the L FIR filter coefficients for the ith-input-to-jth-output channel of the controller. The J controller output signals propagate to the desired quiet region, where K error sensors measure the error signals $\epsilon^{(k)}(n)$, $1 \le k \le K$ as

Equation (21): $$\epsilon^{(k)}(n) = d^{(k)}(n) + \sum_{j=1}^{J} \sum_{m=-\infty}^{\infty} h_m^{(j,k)} y^{(j)}(n-m),$$

and $h_m^{(i,j)}$, $-\infty < m < \infty$ is the jth-output-to-kth-error plant impulse response channel.

In the original filtered-X LMS algorithm, IJK filtered input signals $f^{(i,j,k)}(n)$ are calculated as

Equation (22): $$f^{(i,j,k)}(n) = \sum_{m=1}^{M} h_m^{(j,k)} x^{(i)}(n-m),$$

from which $w_l^{(i,j)}(n)$ is updated as

Equation (23): $$w_l^{(i,j)}(n+1) = w_l^{(i,j)}(n) - \sum_{k=1}^{K} \epsilon_\mu^{(k)}(n) f^{(i,j,k)}(n-l),$$

where

Equation (24): $$\epsilon_\mu^{(k)}(n) = \mu(n)\epsilon^{(k)}(n).$$

A careful study of the filtered-X LMS algorithm described by equations (20), (22), (23), and (24) reveals the fact that this implementation requires $IJK(L+M) + K$ MACs to compute the coefficient updates, even though computing the controller outputs only requires IJL MACs. Thus, the complexity of the update calculations is more than K times the complexity of the input-output calculations. For systems with a large number of error sensors, the computational burden of the coefficient updates can overwhelm the capabilities of the processor chosen for the control task.

The standard implementation of the filtered-X LMS algorithm also has memory requirements that can exceed the capabilities of a chosen processor. The total storage needed is $I\,J(K+1)L + J\,K\,M + I\max(L, M+1) + K$, and for long controller filter lengths, the bulk of this storage is for the $I\,J\,K\,L$ filtered input signals $f^{(i,j,k)}(n-1)$. Clearly, it is desirable to find alternative implementations of the filtered-X LMS

algorithm that have reduced computational and memory requirements.

It is now easier to illustrate the presently preferred embodiment for an algorithm that is based on the method described above for the single channel system.

The preferred embodiment of present invention is a multichannel extension of the new version of the filtered-X LMS algorithm for the single channel control system above. To determine the appropriate grouping of terms for the coerficient updates, the expression for $f^{(i,j,k)}(n-l)$ is substituted in (22) into the update in (23) to get

Equations (25) and (26):

$$w_l^{(i,j)}(n+1) = w_l^{(i,j)}(n) - \sum_{k=1}^{K} \epsilon_\mu^{(k)}(n) \sum_{m=1}^{M} h_m^{(j,k)} x^{(i)}(n-l-m)$$

$$= w_l^{(i,j)}(n) - \sum_{m=1}^{M} \varepsilon_m^{(j)}(n) x^{(i)}(n-l-m),$$

where the $J\ M$ terms $\epsilon_m^{(j)}(n)$ for $1 \le j \le J$ and $1 \le m \le M$ are defined as

Equation (27): $$\varepsilon_m^{(j)}(n) = \sum_{k=1}^{K} h_m^{(j,k)} \epsilon_\mu^{(k)}(n).$$

Comparing (26) with (7), they are seen to be of a similar form. Thus, a method is obtained which is analogous to the implementation of the multichannel filtered-X LMS algorithm. Define

Equation (28): $$e_m^{(j)}(n) = \sum_{p=1}^{m} \varepsilon_p^{(j)}(n-m+p).$$

Then, a set of $I\ J\ L$ auxiliary coefficients $\varpi l^{(i,j)}(n)$ is defined whose updates are given by

Equation (29): $$\hat{w}_l^{(i,j)}(n+1) = \hat{w}_l^{(i,j)}(n) - e_M^{(j)}(n) x^{(i)}(n-M-l).$$

To compute the controller outputs, the multichannel

equivalent of equation (16) is

Equation (30):
$$y^{(j)}(n) = \sum_{i=1}^{I}\sum_{l=0}^{L-1} \hat{w}_l^{(i,j)}(n)x^{(i)}(n-l) - \sum_{m=1}^{M-1} e_m^{(j)}(n-1)r_{m+1}(n),$$

where $r_m(n)$ in this case is defined as

Equation (31):
$$r_m(n) = \sum_{i=1}^{I}\sum_{l=0}^{L-1} x^{(i)}(n-l)x^{(i)}(n-l-m).$$

In analogy with equation (17), $r_m(n)$ can be recursively computed as

Equation (32):
$$r_m(n) = r_m(n-1) + \sum_{i=1}^{I}\left\{x^{(i)}(n)x^{(i)}(n-m) - x^{(i)}(n-L)x^{(i)}(n-L-m)\right\}.$$

Similarly, $e_m^{(j)}(n)$ has an update similar to that in equation (18), as given by Equation (33):

$$e_m^{(j)}(n) = \begin{cases} \sum_{k=1}^{K} h_1^{(j,k)}\epsilon_\mu^{(k)}(n) & \text{if } m = 1 \\ e_{m-1}^{(j)}(n-1) + \sum_{k=1}^{K} h_m^{(j,k)}\epsilon_\mu^{(k)}(n) & \text{if } 2 \le m \le M \end{cases}$$

Collecting equations (24), (29), (30), (32), and (33), the result is an alternative equivalent implementation of the multichannel filtered-X LMS algorithm. Table 1 lists the operations of this implementation, along with the number of MACs required to implement each operation.

TABLE 1

Equation	*MACs*
for $m = 2$ to M do	
$r_m(n) = r_m(n-1) + \sum_{i=1}^{I} \left\{ x^{(i)}(n)x^{(i)}(n-m) - x^{(i)}(n-L)x^{(i)}(n-L-m) \right\}$	$2I(M-1)$
end	
for $k = 1$ to K do	
$\epsilon_\mu^{(k)}(n) = \mu(n)\epsilon^{(k)}(n)$	K
end	
for $j = 1$ to J do	
$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} \hat{w}_l^{(i,j)}(n)x^{(i)}(n-l) - \sum_{m=1}^{M-1} e_m^{(j)}(n-1)r_{m+1}(n)$	$IJL + J(M-1)$
$e_1^{(j)}(n) = \sum_{k=1}^{K} h_1^{(j,k)}\epsilon_\mu^{(k)}(n)$	JK
for $m = 1$ to $M-1$ do	
$e_{m+1}^{(j)}(n) = e_m^{(j)}(n-1) + \sum_{k=1}^{K} h_{m+1}^{(j,k)}\epsilon_\mu^{(k)}(n)$	$JK(M-1)$
end	
for $i = 1$ to I do	
for $l = 0$ to $L-1$ do	
$\hat{w}_l^{(i,j)}(n+1) = \hat{w}_l^{(i,j)}(n) - e_M^{(j)}(n)x^{(i)}(n-M-l)$	IJL
end	
end	
end	
Total: $2IJL + JKM + (2I+J)(M-1) + K$	

The algorithm employs $2IJL + JKM + (2I + J)(M - 1) + K$ **MACs per iteration, and it requires** $IJL + JKM + IL + (I + J + 1)M + K - 1$ **memory locations to implement. IT should be noted that this implementation of the multichannel filtered-X LMS adaptive controller modifies the adjoint LMS/CPFE adaptive controller by including the second summation of the RHS of equation (30) and the supporting updates for** $e_m^{(j)}(n)$ **and** $r_m(n)$**, respectively. Since** $e_m^{(j)}(n)$ **is of** $O(\mu(n))$**, the performance difference between the multichannel filtered-X and adjoint LMS/CPFE algorithms can only be expected to be significant for large step sizes. Because the adjoint LMS/CPFE algorithm**

is a filtered-error technique with an approximate group delay of M samples in the update rule, however, its performance is often worse than that of the filtered-X LMS algorithm. Moreover, the complexity difference between the two algorithms is relatively insignificant for systems with a large number of channels, as will now be shown.

It is useful to compare the computational and memory requirements of the original and fast implementations of the multichannel filtered-X LMS algorithm so as to understand the benefits of the present invention. In this comparison, three different problem scenarios are considered. Each scenario is defined by specific choices of the controller filter length L and plant model filter length M that might be appropriate for a particular type of noise or vibration control task. In each case, the quantities $R_F^{(C)}$ and $R_F^{(M)}$ denote the ratios of the numbers of MACs and memory locations, respectively, required by the fast implementation with respect to the numbers of MACs and memory locations needed for the original implementation. For comparison, the corresponding ratios $R_A(C)$ and $R_A(M)$ are provided for the adjoint LMS/CPFE algorithm with respect to the original filtered-X LMS algorithm. Since the adjoint LMS/CPFE algorithm equations are used within the fast implementation, it is the case where $R_A^{(C)} < R_F^{(C)}$ and $R_A^{(M)} < R_F^{(M)}$, although the two algorithms' requirements are similar for systems with a large number of channels.

The first situation considered is a broadband noise control task in which the controller and plant model filter lengths are $L = 50$ and $M = 25$. These choices offer a reasonable balance between the performance and

robustness of the controller in this situation for fixes hardware resourc s. Table 2 shows the complexity and memory ratios for the different cases considered.

TABLE 2

# Channels			Complexity Ratio		Memory Ratio		# Channels			Complexity Ratio		Memory Ratio	
I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$	I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$
1	2	2	0.7512	0.9900	0.7235	0.7765	4	6	4	0.3574	0.3974	0.3313	0.3348
1	3	2	0.7508	0.9502	0.6933	0.7301	2	6	6	0.3506	0.3906	0.3469	0.3515
1	4	2	0.7506	0.9302	0.6772	0.7054	2	8	6	0.3505	0.3865	0.3412	0.3446
1	6	2	0.7504	0.9101	0.6605	0.6797	2	8	8	0.3082	0.3359	0.3096	0.3123
2	2	2	0.6259	0.8055	0.6259	0.6559	4	8	8	0.2311	0.2495	0.2288	0.2303
2	3	2	0.6256	0.7654	0.5877	0.6085	8	8	8	0.1925	0.2063	0.1820	0.1828
2	4	2	0.6255	0.7453	0.5672	0.5832	4	8	12	0.1845	0.1972	0.1927	0.1937
2	6	2	0.6253	0.7252	0.5459	0.5568	4	12	12	0.1844	0.1949	0.1888	0.1895
1	4	4	0.5726	0.6752	0.5358	0.5523	4	16	12	0.1844	0.1938	0.1869	0.1874
1	6	4	0.5722	0.6635	0.5241	0.5353	8	8	12	0.1449	0.1544	0.1444	0.1449
2	3	3	0.5009	0.6025	0.4928	0.5085	8	8	16	0.1202	0.1274	0.1244	0.1248
1	4	6	0.5015	0.5733	0.4771	0.4888	8	12	16	0.1201	0.1257	0.1211	0.1214
3	3	3	0.4552	0.5424	0.4490	0.4601	8	16	16	0.1201	0.1249	0.1195	0.1197
3	6	3	0.4549	0.5130	0.4111	0.4168	8	32	16	0.1200	0.1236	0.1170	0.1171
2	4	4	0.4294	0.4979	0.4209	0.4305	16	16	16	0.1000	0.1036	0.0926	0.0927
2	6	4	0.4291	0.4862	0.4060	0.4125	16	32	16	0.1000	0.1024	0.0901	0.0901
2	8	4	0.4290	0.4804	0.3985	0.4033	8	16	32	0.0817	0.0842	0.0901	0.0903
4	4	4	0.3576	0.4090	0.3484	0.3536	16	16	32	0.0613	0.0631	0.0625	0.0625
2	4	6	0.3510	0.3989	0.3582	0.3651	32	32	32	0.0510	0.0520	0.0466	0.0466

As can be seen for all of the cases considered, the number of multiples required for the new implementation of the multichannel filtered-X LMS algorithm is less than that of the original algorithm, and this difference is significant for systems with a large number of channels. In fact, for an N-input, N-output, N-error system, the complexity of the new implementation is approximately 80% of the original implementation when $N = 2$, 40% of the original when $N = 4$, 20% of the original when $N = 8$, and 10% of the original when $N = 16$. In addition, the number of memory locations required by the new implementation is

also reduced and is less than 10% of the original algorithm's memory requirements for $I = J = K = N = 16$. These savings are significant, as they allow a multichannel control system to be implemented on a much-simpler hardware platform.

A second scenario is where $L = 2$ and $M = 10$. Such a situation is typical of narrowband noise control problems in which each input signal is a single sinusoid of a different frequency; thus, each channel of the controller is dedicated to one tonal component of the unwanted acoustic field. Table 3 lists the ratio of MACs and memory locations for the two algorithms with respect to the original filtered-X LMS algorithm in this situation.

TABLE 3

# Channels			Complexity Ratio		Memory Ratio		# Channels			Complexity Ratio		Memory Ratio	
I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$	I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$
2	2	2	0.5472	1.0566	1.0682	1.1705	6	12	6	0.1901	0.2306	0.0.5900	0.5950
2	3	2	0.5443	0.9430	1.0667	1.1417	6	6	12	0.1663	0.1970	0.5398	0.5450
2	4	2	0.5429	0.8857	1.0658	1.1250	6	12	12	0.1653	0.1859	0.5274	0.5301
2	2	4	0.4902	0.7549	0.9315	0.9932	8	16	12	0.1307	0.1461	0.4570	0.4587
2	4	4	0.4851	0.6634	0.9173	0.9511	8	12	16	0.1245	0.1380	0.4433	0.4450
2	6	4	0.4834	0.6325	0.9119	0.9352	8	16	16	0.1243	0.1359	0.4401	0.4414
2	6	6	0.4631	0.5638	0.8525	0.8687	8	16	24	0.1178	0.1256	0.4227	0.4236
2	6	8	0.4527	0.5287	0.8209	0.8333	8	16	32	0.1146	0.1204	0.4138	0.4145
4	6	4	0.2824	0.3870	0.7576	0.7746	16	32	16	0.0723	0.0781	0.2937	0.2941
4	8	6	0.2586	0.3193	0.6933	0.7025	16	16	32	0.0625	0.0669	0.2707	0.2711
4	8	8	0.2468	0.2926	0.6625	0.6696	16	32	32	0.0623	0.0652	0.2669	0.2671

As can be seen, except for systems with a small number of channels, the new implementation requires only a fraction of the MACs and memory locations used by the original implementation. Thus, the new implementation reduces the controller's hardware complexity in narrowband control situations as well.

The third problem scenario consid red is a task in which $L = 10$ and $M = 20$. These choices are typical for noise and vibration control tasks in which the input signals are measured by physical sensors, but the primary goal of the controller is to attenuate a relatively few number of tonal components. Table 4 lists the respective complexity and memory usage ratios for different cases.

TABLE 4

# Channels			Complexity Ratio		Memory Ratio		# Channels			Complexity Ratio		Memory Ratio	
I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$	I	J	K	$R_A^{(C)}$	$R_F^{(C)}$	$R_A^{(M)}$	$R_F^{(M)}$
2	2	2	0.5745	0.9787	0.9098	0.9877	6	12	6	0.2109	0.2442	0.3911	0.3940
2	3	2	0.5735	0.8886	0.8779	0.9331	6	6	12	0.1629	0.1886	0.3375	0.3405
2	4	2	0.5730	0.8434	0.8604	0.9032	6	12	12	0.1625	0.1796	0.3257	0.3273
2	2	4	0.4656	0.6832	0.7488	0.7956	8	16	12	0.1354	0.1482	0.2755	0.2764
2	4	4	0.4636	0.6092	0.7102	0.7350	8	12	16	0.1227	0.1341	0.2603	0.2613
2	6	4	0.4629	0.5844	0.6963	0.7131	8	16	16	0.1227	0.1324	0.2578	0.2585
2	6	6	0.4226	0.5057	0.6381	0.6499	8	16	24	0.1098	0.1163	0.2394	0.2398
2	6	8	0.4016	0.4648	0.6067	0.6158	8	16	32	0.1033	0.1082	0.2299	0.2303
4	6	4	0.3086	0.3937	0.5452	0.5560	16	32	16	0.0817	0.0865	0.1690	0.1692
4	8	6	0.2639	0.3138	0.4760	0.4818	16	16	32	0.0620	0.0656	0.1434	0.1436
4	8	8	0.2408	0.2787	0.4440	0.4485	16	32	32	0.0619	0.0644	0.1409	0.1410

As in the previous cases, the new implementation of the filtered-X LMS algorithm save computations and memory locations for systems with a large number of channels.

It is now necessary to discuss the plant delay on the filtered-X LMS algorithm's operation and the resulting LMS algorithm for active noise control. Considering the single-channel filtered-X LMS adaptive controller, it is seen from equation (2) that the error signal $\epsilon(n)$ depends on the outputs $y(n - m)$ of the controller at different time instants, which in turn depend on the controller coefficients $w_1(n - m)$ at

different time instants. Because the plant is typically causal, past coefficients are employed within the gradient-based updates, causing a decrease in the performance of the system not unlike that observed for the delayed LMS algorithm. It is possible to largely mitigate the effects of this delay by computing a delay-compensated error signal that depends on the most-recent coefficients $w_l(n)$. Figure 2 shows the block diagram of this system, in which $\gamma(k)$ is the delay-compensated error signal given by

Equation (34):
$$\gamma(n) = \left\{\epsilon(n) - \sum_{m=1}^{M} h_m y(n-m)\right\} + \sum_{l=0}^{L-1} w_l(n) f(n-l),$$

where the term within brackets on the RHS of (34) is nearly the same as the unattenuated noise signal $d(n)$ if the estimated impulse response h_m accurately models the unknown plant's impulse response. The LMS algorithm for active noise control uses $\gamma(k)$ to update the coefficients as [20, 21]

Equation (35):
$$w_l(n+1) = w_l(n) - \mu(k)\gamma(k)f(n-l).$$

This algorithm requires a total of $3L + 2M + 1$ MACs per iteration to implement, and it uses $2L + M + \max\{L, M + 1\} + 1$ memory locations. Note that this algorithm's performance depends on how well the estimated plant impulse response models the physical response of the plant. As our focus is on implementation and not performance issues, a performance analysis of the multichannel LMS algorithm for active noise control is beyond the scope of this paper.

It is possible to extend the above algorithm to the multichannel case. In this situation, the K delay-

compensated error signals are calculated as

Equation (36):

$$\gamma^{(k)}(n) = \epsilon^{(k)}(n) - \sum_{j=1}^{J}\sum_{m=1}^{M} h_m^{(j,k)} y^{(j)}(n-m) + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{l=0}^{L-1} w_l^{(i,j)}(n) f^{(i,j,k)}(n-l),$$

at which point $\mu(n)\gamma^{(k)}(n)$ is used in place of $\epsilon\mu^{(k)}(n)$ in (23). Unfortunately, this modification adds $I\ J\ K\ L + J\ K\ M$ MACs per iteration to the overall requirements of the adaptive system if the necessary $f^{(i,j,k)}(n)$ values are available, and it adds $I\ J\ K\ L + (2I + 1)\ J\ K\ M$ MACs if $f^{(i,j,k)}(n)$ must be computed. In addition, the storage requirements for the overall system are significantly increased if the modification is applied to the fast multichannel filtered-X LMS algorithm in Table 1.

In the prior art, a method is presented for reducing the complexity of the single-channel LMS algorithm for active noise control when the secondary path length M is less than a third of the controller filter length L. This algorithm is now extended to the multichannel case. Define

Equation (37): $\gamma_\mu^{(n)}(n) = \mu(n)\gamma^{(k)}(n) = \mu(n)\left\{\epsilon^{(k)}(n) + \sum_{j=1}^{J}\sum_{m=1}^{M} h_m^{(j,k)} u_m^{(j)}(n-1)\right\},$

where $u_m^{(j)}(n)$ is defined as

Equation (38):

$$u_m^{(j)}(n) = y^{(j)}(n-m) - \sum_{i=1}^{I}\sum_{l=0}^{L-1} w_l^{(i,j)}(n+1)x^{(i)}(n-l-m).$$

Using algebraic manipulations similar to those in the prior art, an update for $u_m^{(j)}(n)$ is found to be

Equation (39):

$$u_m^{(j)}(n) = \begin{cases} -\sum_{k=1}^{K} \rho_0^{(j,k)}(n)\gamma_\mu^{(k)}(n) & \text{if } m = 1 \\ u_{m-1}^{(j)}(n-1) - \sum_{k=1}^{K} \rho_{m-1}^{(j,k)}(n)\gamma_\mu^{(k)}(n) & \text{if } 2 \le m \le M \end{cases},$$

where $\rho_m^{(j,k)}(n)$ is defined as

Equation (40): $$\rho_m^{(j,k)}(n) = \sum_{i=1}^{I}\sum_{l=0}^{L-1} x^{(i)}(n-l-m)f^{(i,j,k)}(n-l).$$

Note that $\rho_m^{(j,k)}(n)$ can be updated as

Equation (41): $$\rho_m^{(j,k)}(n) = \rho_m^{(j,k)}(n-1)+\sum_{i=1}^{I}\left\{x^{(i)}(n-m)f^{(i,j,k)}(n)-x^{(i)}(n-m-L)f^{(i,j,k)}(n-L)\right\},$$

which greatly reduces the number of operations needed for the algorithm when L is large. This update also reduces the amount of memory required for the algorithm, as $f^{(i,j,k)}(n)$ and $f^{(i,j,k)}(n-L)$ can be computed at each iteration to avoid storing $f^{(i,j,k)}(n-l)$ for $1 \le l \le L-1$.

Collecting equations (37), (39), and (41), a multichannel delay compensation technique is obtained that requires $(4I+2)\,J\,K\,M$ MACs per iteration and $(I+J+J\,K)M$ memory locations to implement when $L > M$, assuming that $f^{(i,j,k)}(n)$ and $f^{(i,j,k)}(n-L)$ are computed at each iteration. Comparing these complexity requirements with those of the original delay compensation technique, if

Equation (42): $$L > \left(2+\frac{1}{I}\right)M,$$

then this new technique is more computationally efficient. The new technique also has low memory requirements and thus is an ideal match to the fast algorithm in Table 1.

Although useful, the delay-compensation method in equations (37), (39), and (41) can be prohibitive to implement when the number of channels is large, as its complexity grows as $O\,(I\,J\,K\,M)$.

There is an alternate implementation that uses many of the existing quantities within the efficient

multichannel filtered-X LMS algorithm in Table 1 while avoiding the formation of the filtered input signal values. For this derivation, consider the definition of $\rho_m^{(j,k)}(n)$ in (40). Substituting the expression for $f^{(i,j,k)}(n)$ in (22) into the RIIS of (40) and rearranging terms, the result is Equation (43):

$$\rho_m^{(j,k)}(n) = \sum_{q=1}^{M} h_q^{(j,k)} \left\{ \sum_{l=0}^{L-1} \sum_{i=1}^{I} x^{(i)}(n-l-m) x^{(i)}(n-l-q) \right\} = \sum_{q=1}^{M} h_q^{(j,k)} r_{m-q}(n-q),$$

where $r_m(n)$ is as defined in (32). From the definition of $r_m(n)$, it is straightforward to show that

Equation (44): $$r_{m-q}(n-q) = r_{q-m}(n-m),$$

and thus the necessary values of $r_{m-q}(n-q)$ to represent $\rho_m^{(j,k)}(n)$ can be obtained from $r_m(n)$, $0 \le m \le M$ by storing delayed values of these quantities. Define

Equation (45): $$c_m^{(j)}(n) = \sum_{k=1}^{K} h_m^{(j,k)}(n) \gamma_\mu^{(k)}(n).$$

Note that $c_m^{(j)}(n)$ appears in the update for $e_m^{(j)}(n)$ in (33) when the delay compensation technique is combined with the fast filtered-X LMS algorithm; thus, it is already available. Then,

Equation (46): $$\sum_{k=1}^{K} \rho_{m-1}^{(j,k)}(n) \gamma_\mu^{(k)}(n) = \sum_{q=1}^{M} c_q^{(j)}(n) r_{m-1-q}(n-q),$$

and the RHS of (46) can replace the summations of the RIIS of the updates for $u_m^{(j)}(n)$ in (39).

Table 5 lists the operations for this alternative form of the LMS algorithm for multichannel active noise.

TABLE 5

Equation	*MACs*
for $m = 0$ to M do	
$r_m(n) = r_m(n-1) + \sum_{i=1}^{I} \left\{ x^{(i)}(n)x^{(i)}(n-m) - x^{(i)}(n-L)x^{(i)}(n-L-m) \right\}$	$2I(M+1)$
end	
for $k = 1$ to K do	
$\gamma_\mu^{(k)}(n) = \mu(n) \left\{ \epsilon^{(k)}(n) + \sum_{j=1}^{J} \sum_{m=1}^{M} h_m^{(j,k)} u_m^{(j)}(n-1) \right\}$	$JKM + K$
end	
for $j = 1$ to J do	
$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} \hat{w}_l^{(i,j)}(n) x^{(i)}(n-l) - \sum_{m=1}^{M-1} e_m^{(j)}(n-1) r_{m+1}(n)$	$IJL + J(M-1)$
$c_1^{(j)}(n) = \sum_{k=1}^{K} h_1^{(j,k)} \gamma_\mu^{(k)}(n)$	JK
$e_1^{(j)}(n) = c_1^{(j)}(n)$	
for $m = 1$ to $M-1$ do	
$c_{m+1}^{(j)}(n) = \sum_{k=1}^{K} h_{m+1}^{(j,k)} \gamma_\mu^{(k)}(n)$	$JK(M-1)$
$e_{m+1}^{(j)}(n) = e_m^{(j)}(n-1) + c_{m+1}^{(j)}(n)$	JM
end	
$u_1^{(j)}(n) = -\sum_{q=1}^{M} c_q^{(j)}(n) r_q(n)$	JM
for $m = 1$ to $M-1$ do	
$u_{m+1}^{(j)}(n) = u_m^{(j)}(n-1) - \sum_{q=1}^{M} c_q^{(j)}(n) r_{m-q}(n-q)$	$JM(M-1)$
end	
for $i = 1$ to I do	
for $l = 0$ to $L-1$ do	
$\hat{w}_l^{(i,j)}(n+1) = \hat{w}_l^{(i,j)}(n) - e_M^{(j)}(n) x^{(i)}(n-M-l)$	IJL
end	
end	
end	
Total: $2IJL + 2JKM + (2I + JM)(M+1) + K$	

This algorithm requires $J\ K\ M + J(M+1)\ (M-1)$ **more**

MACs per iteration than does the filtered-X LMS algorithm in Table 1. If

Equation (47): $M - \frac{1}{M} < (4I+1)K,$

then this implementation is more computationally-efficient than that in equations (37), (39), and (41). If

Equation (48): $M < IK + \sqrt{IKL + I^2K^2 + 1},$

then this implementation is more computationally-efficient than the standard implementation in (36). Considering the system configurations listed in Tables 2, 3, and 4, it is observed that the algorithm in Table 5 is the most-computationally-efficient method out of the three delay-compensation techniques considered when $I\,K \geq 7$, $I\,K \geq 5$, and $I\,K \geq 8$, respectively. For the remaining configurations, the standard delay-compensation implementation combined with the new filtered-X LMS update method in Table 2 is the most efficient, although the method in (37), (39), and (41) is the most efficient for the configurations in Table 2 if the controller filter length is increased to $L = 75$. Note that this implementation of the multichannel LMS adaptive controller modifies the filtered-X LMS adaptive controller by including the summation within brackets of the RHS of (37) and the supporting updates for $u_m^{(j)}(n)$. Since $u_m^{(j)}(n)$ is of $O(\mu(n))$, the performance difference between the multichannel LMS and filtered-X LMS algorithms can only be expected to be significant for large step sizes. Also note that the filtered-X LMS algorithm is typically derived assuming "slow adaptation," so that the derivatives of the error signals

with respect to the filter coefficients can be easily calculated. Our multichannel LMS algorithms quantitatively define the difference between the filtered-X LMS and LMS coefficient updates and provide an alternative justification for the former algorithm for situations in which the step size is small-valued.

Next, the effects that numerical errors due to finite precision calculations have on the performances of the new implementation of the filtered-X LMS and LMS algorithms for active noise control are considered. One important feature of the LMS algorithm in adaptive filtering is its robust behavior in the presence of various approximations and errors that are often introduced in a real-world implementation. Since the original implementation of the filtered-X LMS algorithm and the adjoint LMS/CPFE algorithm are variants of stochastic gradient methods, they share many of the robust convergence properties of the LMS algorithm. The new implementations of the filtered-X LMS and LMS algorithms apply one or more forms of delay compensation to the adjoint LMS/CPFE algorithm. As such, the numerical properties of the delay compensation techniques are of immediate interest, particularly as they affect the long-term performances of the systems.

Extensive simulation of the implementations have indicated that the robust numerical properties of the underlying stochastic gradient algorithms are not fundamentally altered in our new implementations. These behaviors are quite unlike those of fast RLS/Kalman techniques that exhibit an exponential instability unless careful measures are taken. The only possible source of numerical difficulty is the method for calculating $r_m(n)$

in (32), as this update is marginally-stable. Thus, numerical errors in $r_m(n)$ can grow linearly over tim in a finite-precision environment. Fortunately, the growth in these errors can be easily prevented using several well-known procedures. Perhaps the simplest procedure is to periodically recalculate $r_m(n)$ using its definition in (31), a procedure that requires extra additions and memory locations. Moreover, because each $r_m(n)$ has a finite memory by definition, accumulating and copying its value to the appropriate memory location within the controller causes no performance penalty, unlike periodic restart methods in exponentially-windowed fast RLS/Kalman filters. Another solution is to introduce a leakage factor into the calculation of $r_m(n)$. One particularly-useful method, described in more detail in [33], is

Equation (49):

$$r_m(n) = \begin{cases} \lambda\left\{r_m(n-1) - \sum_{i=1}^{I} x^{(i)}(n-L)x^{(i)}(n-L-m)\right\} + \sum_{i=1}^{I} x^{(i)}(n)x^{(i)}(n-m) \\ \qquad \text{if } n \bmod L = 0 \\ r_m(n-1) + \sum_{i=1}^{I} x^{(i)}(n)x^{(i)}(n-m) - \lambda\left\{\sum_{i=1}^{I} x^{(i)}(n-L)x^{(i)}(n-L-m)\right\} \\ \qquad \text{otherwise,} \end{cases}$$

where λ is slightly less than one. This method alters the value of $r_m(n)$ slightly, but for values of λ close to one, the errors introduced into the calculations for $y(n)$ do not significantly affect the overall behaviors of the respective systems. Moreover, the update in equation (49) adds only M MACs and a single comparison to each systems' overall complexity.

Figure 3 plots the envelope of the sum-of-squared errors $\sum_{k}^{K} = 1\left\{\epsilon(k)(n)\right\}^2$ of seven different four-input, three-output, four-error active noise control algorithms

with $L = M = 50$ as applied to air compressor data measured in an anechoic environment. In this case, all calculations were performed in the MATLAB floating-point environment, and the approximate sampling rate of the data was 4kHz. Step sizes for each algorithm were chosen to provide the fastest convergence on this data while yielding approximately the same steady-state error powder due to limits in noise modeling error. Figure 3(a) shows the unattenuated air compressor noise signal, in which the bursty nature of the compressor noise is clearly evident, along with the average error power envelopes of the original filtered-X LMS and LMS algorithms applied to this data, in which the step sizes for each algorithm were chosen as $\mu = 0.1$ and $\mu = 0.2$, respectively. Shown for comparison in Figure 3(b) are the average error power envelopes of the adjoint LMS/CPFE algorithm, the fast-filtered-X LMS algorithm in Table 1, and the new multichannel LMS algorithm in table 5, in which the step sizes for each algorithm were chosen as $\mu = 0.05$, $\mu = 0.1$, and $\mu = 0.2$, respectively. As can be seen, the fast multichannel filtered-X LMS algorithm out-performs the adjoint LMS/CPFE algorithm in its convergence rate, and the multichannel LMS algorithm performs the best of the three due to the lack of coefficient delay within the parameter updates. In addition, the differences in the error signals between the original and fast algorithms in Figure 3(a) and (b) were found to be about ten time the order of the machines precision used in the simulation ($\approx 10^{-16}$) after 60000 iterations. A linear growth of the numerical errors was apparent, however.

Shown in Figure 3(c) are the behaviors of the fat multichannel filtered-X LMS and fast multichannel LMS

algorithms in which the leakage-based update for $r_m(n)$ in (49) is employed, where $\lambda = 0.999$. Comparing the average error powers with those of the corresponding algorithms in Figure 3(b), no discernible differences in performance can be seen. In fact, the actual differences between the e5rrors of the corresponding systems were less than 2×10^{-10} in magnitude in this example-a negligible difference-and no growth in the numerical errors of $r_m(n)$ was observed. Thus, the method in (49) can be used to stabilize the marginal instability of the sliding-window $r_m(n)$ updates without altering the observed performances of the proposed systems.

CLAIMS

What is claimed is:

1. A method for actively suppressing a noise at at least one selected location using a multichannel controller system, said method comprising the steps of:

(1) implementing an algorithm in the multichannel controller system to thereby generate a canceling signal to suppress the noise at the at least one selected location;

(2) generating a delay in a feedback loop of the algorithm; and

(3) compensating for the delay in the feedback loop to thereby reduce implementation complexity of the algorithm, and to realize the multichannel controller system which is computationally less expensive than a system which implements an algorithm without the delay in the feedback loop and the delay compensation.

2. The method as defined in claim 1 wherein the method further comprises the step of using an algorithm which is one of a filtered-X LMS algorithm and an LMS algorithm.

3. The method as defined in claim 2 wherein the method further comprises the step of introducing a compensation term in an actuator output signal equation to thereby compensate for the delay in the feedback loop.

4. The method as defined in claim 3 wherein the method further comprises the step of calculating the actuator output signal of the multichannel controller system according to an equation

$$y^{(j)}(n) = \sum_{i=1}^{I}\sum_{l=0}^{L-1} \hat{w}_l^{(i,j)}(n)x^{(i)}(n-1) - \sum_{m=1}^{M-1} e_m^{(j)}(n-1)r_{m+1}(n) \quad ,$$

where $\varepsilon_\mu^{(k)}(n) = \mu(n)\varepsilon^{(k)}(n)$,

where $\hat{w}_l^{(i,j)}(n+1) = \hat{w}_l^{(i,j)}(n) - e_M^{(j)}(n)x^{(i)}(n-M-l).$

where

$$r_m(n) = r_m(n-1) + \sum_{i=1}^{I}\left\{x^{(i)}(n)x^{(i)}(n-m) - x^{(i)}(n-L)x^{(i)}(n-L-m)\right\}$$

and where

$$e_m^{(j)}(n) = \begin{cases} \sum_{k=1}^{K} h_1^{(j,k)}\epsilon_\mu^{(k)}(n) & \text{if } m = 1 \\ e_{m-1}^{(j)}(n-1) + \sum_{k=1}^{K} h_m^{(j,k)}\epsilon_\mu^{(k)}(n) & \text{if } 2 \le m \le M \end{cases}$$

5. The method as defined in claim 2 wherein the method further comprises the step of reducing effects of plant delay on the filtered-X LMS algorithm's operation and a resulting LMS algorithm for active noise control.

6. The method as defined in claim 5 wherein the method further comprises the step of using a delay-compensated error signal given by

$$\gamma(n) = \left\{\epsilon(n) - \sum_{m=1}^{M} h_m y(n-m)\right\} + \sum_{l=0}^{L-1} w_l(n)f(n-l),$$

where the term within brackets is approximately an unattenuated noise signal $d(n)$ if an estimated impulse response h_m accuratley models an unknown impulse response of the plant.

7. The method as defined in claim 6 wherein the method further comprises the step of updating coefficients utilizing an LMS algorithm for active noise control by

the equation

$$w_l(n+1) = w_l(n) - \mu(k)\gamma(k)f(n-l).$$

8. The method as defined in claim 1 wherein the method further comprises the step of applying the principles of the invention to vibration control.

9. A method for actively suppressing noise at at least one selected location using a single channel controller system, said method comprising the steps of:

(1) implementing a filtered-X LMS algorithm in the single channel controller system to thereby generate a canceling signal to suppress the noise at the at least one selected location;

(2) generating a delay in a feedback loop; and

(3) compensating for the delay in the feedback loop to thereby reduce implementation complexity of the filtered-X LMS algorithm, and to realize the single channel controller system which is computationally less expensive than a system which implements a filtered-X LMS algorithm without the delay in the feedback loop and the delay compensation.

10. The method as defined in claim 9 wherein the method further comprises the step of introducing a compensation term in an actuator output signal equation to thereby compensate for the delay in the feedback loop.

11. The method as defined in claim 10 wherein the method

furth r compris s th st p of calculating the actuator output signal of the single channel controller system according to an equation which includes a second summation which is the compensation term, said equation being

$$y(n) = \sum_{l=0}^{L-1} \hat{w}_l(n)x(n-l) - \sum_{m=1}^{M-1} e_m(n-1)r_{m+1}(n)$$

where $\hat{w}_l(n+1) = \hat{w}_l(n) - e_M(n)x(n-M-l)$, and

where $r_m(n) = r_m(n-1) + x(n)x(n-m) - x(n-L)x(n-L-m)$,

where $e_m(n) = \quad h_1 \in \mu(n) \cdot if \cdot m = 1$

$$e_{m-1}(n-1) + h_m \in \mu(n) \cdot if\ 2 \le m \le M$$, and

where $\in \mu(n) = \mu(n)c(n)$.

12. A method for actively suppressing noise in at least one acoustic region by superimposing an equal-but-opposite acoustical field in the at least one acoustic region by using a control system, said method comprising the steps of:

(1) measuring in real time a source of the noise using at least one acoustical sensor to thereby generate data;

(2) digitally processing the data to generate at least one canceling signal;

(3) transmitting the at least one canceling signal to the at least one acoustic region using an acoustical actuator;

(4) providing at least one error sensor in the acoustic region to thereby provide feedback in a feedback loop regarding to the control system;

(5) introducing a delay into the feedback loop to thereby reduce the complexity of calculations of the control system; and

(6) compensating for the delay in the feedback loop to thereby obtain a control system which requires less memory space because of the reduced complexity of the calculations.

13. The method as defined in claim 12 wherein the method further comprises the step of utilizing a filtered-X LMS algorithm in the control system to thereby generate the at least one canceling signal.

14. The method as defined in claim 12 wherein the method further comprises the step of eliminating a signal recombination step that is computationally costly in the feedback loop.

15. The method as defined in claim 12 wherein the method further comprises the step of modifying a method of calculating LMS coefficient updates in the filtered-X LMS algorithm, wherein the modified method is possible because of the delay introduced in the feedback loop, and wherein the modified method is computationally less complex than an unmodified filtered-X LMS algorithm.

16. The method as defined in claim 12 wherein the method further comprises the steps of:

(1) providing a single channel control system; and

(2) calculating an actuator output signal of the single channel controller system according to an equation

$$y(n) = \sum_{l=0}^{L-1} \hat{w}_l(n)x(n-l) - \sum_{m=1}^{M-1} e_m(n-1)r_{m+1}(n)$$

where $\hat{w}_l(n+1) = \hat{w}_l(n) - e_M(n)x(n-M-l)$, and

where $r_m(n) = r_m(n-1) + x(n)x(n-m) - x(n-L)x(n-L-m)$,

where $e_m(n) = \quad h_l \in \mu(n) \cdot if \cdot m = 1$

$e_{m-1}(n-1) + h_m \in \mu(n) \cdot if\ 2 \le m \le M$, and

where $\in \mu(n) = \mu(n)c(n)$.

17. The method as defined in claim 12 wherein the method further comprises the steps of:

(1) providing a multichannel control system; and

(2) calculating an actuator output signal of the multichannel controller system according to an equation

$$y^{(j)}(n) = \sum_{i=1}^{I} \sum_{l=0}^{L-1} \hat{w}_l \cdot (n)x \cdot (n-1) - \sum_{m=1}^{M-1} e_m(n-1)r_{m+1}(n) \quad ,$$

where $\varepsilon_\mu(n) = \mu(n)\varepsilon \cdot (n)$,

where $\hat{w}_l^{(i,j)}(n+1) = \hat{w}_l^{(i,j)}(n) - e_M^{(j)}(n)x^{(i)}(n-M-l)$,

where $r_m(n) = r_m(n-1) + \sum_{i=1}^{I} \left\{ x^{(i)}(n)x^{(i)}(n-m) - x^{(i)}(n-L)x^{(i)}(n-L-m) \right\}$,

and where $e_m^{(j)}(n) = \begin{cases} \sum_{k=1}^{K} h_1^{(j,k)} \epsilon_\mu^{(k)}(n) & \text{if } m = 1 \\ e_{m-1}^{(j)}(n-1) + \sum_{k=1}^{K} h_m^{(j,k)} \epsilon_\mu^{(k)}(n) & \text{if } 2 \le m \le M \end{cases}$.

APPENDIX A 1/6

```
function [e,W] = fims1(mu,L,H,x,d);

% This MATLAB program implements the fast LMS algorithm for
% single-channel active noise control.

% Copyright 1996 by Scott C. Douglas. All Rights Reserved.

M = length(H);
W = zeros(L,1);
X = zeros(L+M,1);
F = zeros(L+1,1);
Y = zeros(M,1);
U = Y;
Rxf = Y;
ntr = length(x);
e = zeros(1,ntr);

for n=1:ntr;

        F(2:L+1) = F(1:L);
        F(1) = X(1:M)'*H;
        X(2:L+M) = X(1:L+M-1);
        X(1) = x(n);

        ynew = Y'*H;
        Y(2:M) = Y(1:M-1);
        Y(1) = W'*X(1:L);
        epsi = d(n) + ynew;

        e(n) = epsi - H'*U;

        Rxf = Rxf + X(1:M)*F(1) - X(L+1:L+M)*F(L+1);

        U = [0; U(1:M-1)] + Rxf*(mu*e(n));

        W = W - mu*e(n)*F(1:L);

end;
```

APPENDIX A 2/6

```
function [e,W] = lms1(mu,L,H,x,d);

% This program implements the LMS algorithm for a single-input,
% single-output system. It is used to check the correctness
% of the fast LMS algorithm for single-channel active noise
% control.

% Copyright 1996 by Scott C. Douglas. All Rights Reserved.

M = length(H);
W = zeros(L,1);
X = zeros(M,1);
F = zeros(L,1);
ntr = length(x);
e = zeros(1,ntr);

for n=1:ntr;

        F(2:L) = F(1:L-1);
        F(1) = H'*X;
        X(2:M) = X(1:M-1);
        X(1) = x(n);

        e(n) = d(n) + W'*F;
        W = W - mu*e(n)*F;

end;
```

APPENDIX A 3/6

```
function [e,W] = ffxlms(mu,L,H,x,d);

% This MATLAB program implements the fast filtered-X LMS algorithm
% for multichannel active noise control.

% Copyright 1996 by Scott C. Douglas. All Rights Reserved.

[Nx,ntr]=size(x);
[Ne,t]=size(d);
[M,t]=size(H);
Ny=t/Ne;
W = zeros(Nx*L,Ny);
Y = zeros(M,Ny);
E = zeros(M,Ny);
R = zeros(M-1,1);
X = zeros(L+M,Nx);
e = zeros(Ne,ntr);

for n=1:ntr;

        epsi = d(:,n);

        for j = 1:Ny;
                epsi = epsi + H(:,(j-1)*Ne+1:(j*Ne))*Y(:,j);
        end;

        for i = 1:Nx;
                R = R + x(i,n)*X(2:M,i) - X(L,i)*X(L+2:L+M,i);
        end;

        X(2:L+M,:) = X(1:L+M-1,:);
        X(1,:) = x(:,n)';

        Y(2:M,:) = Y(1:M-1,:);
        Y(1,:) = -R'*E(1:M-1,:);
        for i = 1:Nx;
                Y(1,:) = Y(1,:) + X(1:L,i)'*W((i-1)*L+1:(i*L),:);
        end;
        for j=1:Ny;
                E(:,j) = [0; E(1:M-1,j)] + H(:,(j-1)*Ne+1:(j*Ne))*(mu*epsi);
        end;

        for i=1:Nx;
                W((i-1)*L+1:(i*L),:) = W((i-1)*L+1:(i*L),:) - X(M+1:M+L,i)*E(M,:);
        end;

        e(:,n) = epsi;

end;
```

APPENDIX A 4/6

```
function [e,W] = fxlms(mu,L,H,x,d);

% This MATLAB program implements the filtered-X LMS algorithm for
% multichannel active noise control. It is used to check the
% correctness of the fast filtered-X LMS algorithm for multichannel
% active noise control.

% Copyright 1996 by Scott C. Douglas. All Rights Reserved.

[Nx,ntr]=size(x);
[Ne,t]=size(d);
[M,t]=size(H);
Ny=t/Ne;
W = zeros(Nx*L,Ny);
F = zeros(Nx*L,Ny*Ne);
Y = zeros(M,Ny);
E = zeros(M,Ny);
X = zeros(L+M+2,Nx);
e = zeros(Ne,ntr);

for n=1:ntr;

    epsi = d(:,n);

    for i = 1:Nx;
        l = [(i-1)*L+1:(i*L)]';
        for j = 1:Ny;
            m = (j-1)*Ne+1:(j*Ne);
            F(l(2:L),m) = F(l(1:L-1),m);
            F(l(1),m) = X(1:M,i)'*H(:,m);
        end;
    end;

        X(2:L+M+2,:) = X(1:L+M+1,:);
        X(1,:) = x(:,n)';

        for j = 1:Ny;
                m = (j-1)*Ne+1:(j*Ne);
                t = H(:,m)'*Y(:,j);
                epsi = epsi + t;
        end;

        e(:,n) = epsi;
        epsi = mu*epsi;

        Y(2:M,:) = Y(1:M-1,:);
        Y(1,:) = zeros(1,Ny);
        for i = 1:Nx;
                l = (i-1)*L+1:(i*L);
                Y(1,:) = Y(1,:) + X(1:L,i)'*W(l,:);
        end;

    for i = 1:Nx;
        l = (i-1)*L+1:(i*L);
        for j = 1:Ny;
            m = (j-1)*Ne+1:(j*Ne);
            W(l,j) = W(l,j) - F(l,m)*epsi;
        end;
    end;

end;
```

APPENDIX A 5/6

```
function [e,W] = fims(mu,L,H,x,d);

% This MATLAB program implements the fast LMS algorithm for
% multichannel active noise control.

% Copyright 1996 by Scott C. Douglas. All Rights Reserved.

[Nx,ntr]=size(x);
[Ne,t]=size(d);
[M,t]=size(H);
Ny=t/Ne;
W = zeros(Nx*L,Ny);
F = zeros(Nx*L,Ny*Ne);
Y = zeros(M,Ny);
E = zeros(M,Ny);
R = zeros(M-1,1);
Rxf = zeros(M,Ny*Ne);
U = zeros(M,Ny);
X = zeros(L+M+2,Nx);
e = zeros(Ne,ntr);
epsi = zeros(Ne,1);
for n=1:ntr;
        for i = 1:Nx;
                R = R + x(i,n)*X(2:M,l) - X(L,l)*X(L+2:L+M,l);
        end;
        X(2:L+M+2,:) = X(1:L+M+1,:);
        X(1,:) = x(:,n)';
        epsi = d(:,n); epsinew = epsi;
        for j = 1:Ny;
                m = (j-1)*Ne+1:(j*Ne);
                for i = 1:Nx;
                        Rxf(:,m) = Rxf(:,m) + X(1:M,i)*(X(2:M+1,i)'*H(:,m)) -
X(L+1:L+M,l)*(X(L+2:L+M+1,l)'*H(:,m));
                end;
                t = H(:,m)'*Y(:,j);
                epsinew = epsinew + t - H(:,m)'*U(:,j);
                epsi = epsi + t;
        end;
        e(:,n) = epsinew;
        epsinew = mu*epsinew;
        for j = 1:Ny;
                m = (j-1)*Ne+1:(j*Ne);
                U(:,j) = [0; U(1:M-1,j)] + Rxf(:,m)*epsinew;
        end;

        Y(2:M,:) = Y(1:M-1,:);
        Y(1,:) = -R'*E(1:M-1,:);
        for i = 1:Nx;
                l = (i-1)*L+1:(i*L);
                Y(1,:) = Y(1,:) + X(1:L,i)'*W(l,:);
        end;
        for j=1:Ny;
                m = (j-1)*Ne+1:(j*Ne);
                E(:,j) = [0; E(1:M-1,j)] + H(:,m)*epsinew;
        end;
        for i=1:Nx;
                l = (i-1)*L+1:(i*L);
                W(l,:) = W(l,:) - X(M+1:M+L,i)*E(M,:);
        end;

end;
```

APPENDIX A 6/6

```
function [e,W] = lms(mu,L,H,x,d);

% This MATLAB program implements the LMS algorithm for
% multichannel active noise control. It is used to check
% the correctness of the fast LMS algorithm for multichannel
% active noise control.

% Copyright 1996 by Scott C. Douglas. All Rights Reserved.

[Nx,ntr]=size(x);
[Ne,t]=size(d);
[M,t]=size(H);
Ny=t/Ne;
W = zeros(Nx*L,Ny);
F = zeros(Nx*L,Ny*Ne);
Y = zeros(M,Ny);
X = zeros(L,Nx);
e = zeros(Ne,ntr);

for n=1:ntr;

        epsi = d(:,n);

        for i = 1:Nx;
                l = [(i-1)*L+1:(i*L)]';
                for j = 1:Ny;
                        m = (j-1)*Ne+1:(j*Ne);
                        F(l(2:L),m) = F(l(1:L-1),m);
                        F(l(1),m) = X(1:M,l)'*H(:,m);
                        epsi = epsi + F(l,m)'*W(l,j);
                end;
        end;

        X(2:L,:) = X(1:L-1,:);
        X(1,:) = x(:,n)';

        e(:,n) = epsi;
        epsi = mu*epsi;

        for i = 1:Nx;
                l = (i-1)*L+1:(i*L);
                for j = 1:Ny;
                        m = (j-1)*Ne+1:(j*Ne);
                        W(l,j) = W(l,j) - F(l,m)*epsi;
                end;
        end;

end;
```

1/2

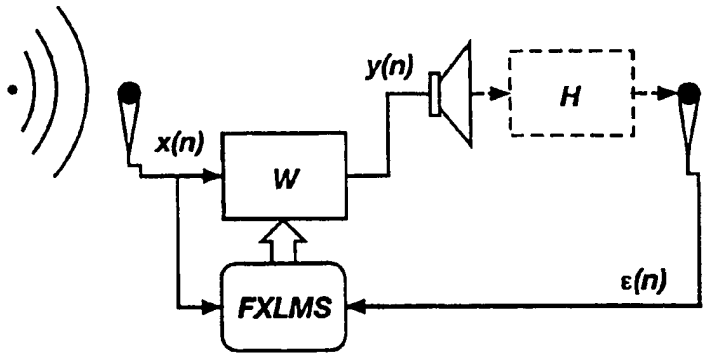

Fig. 1

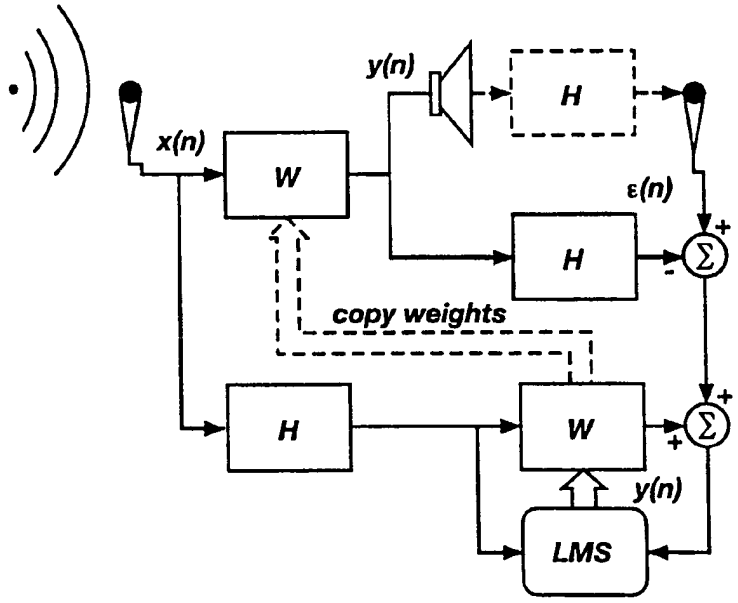

Fig. 2

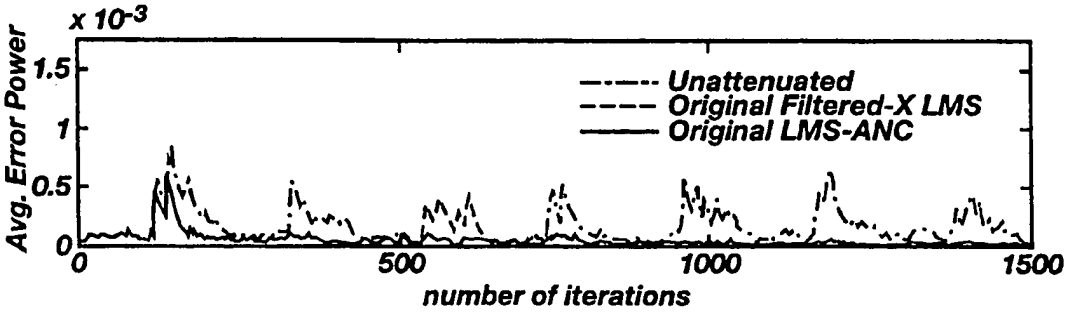

Fig. 3(a)

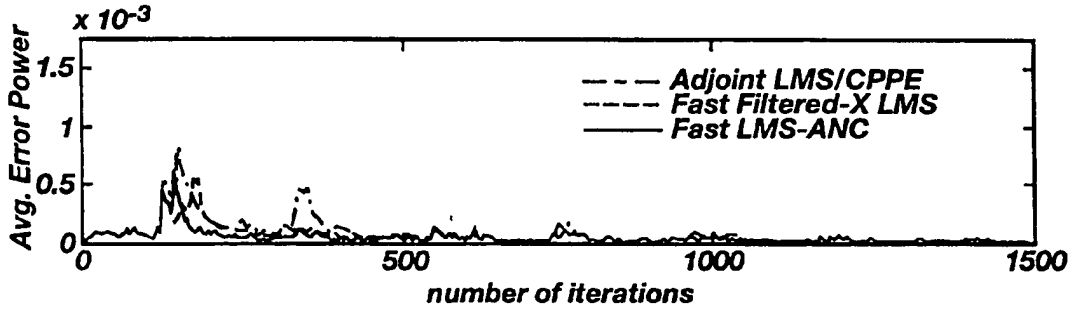

Fig. 3(b)

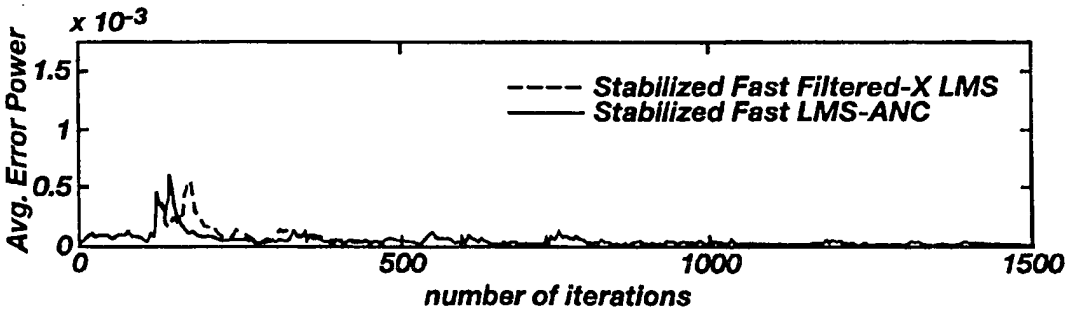

Fig. 3(c)